UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

Trabajo final del Grado en Ingeniería Informática:

Aplicabilidad de Gazebo en la simulación de vehículos autónomos

Presentado por Carmelo Basconcillos Reyero
en julio de 2017
Tutor D. Jesús Enrique Sierra García

UNIVERSIDAD DE BURGOS
ESCUELA POLITÉCNICA SUPERIOR
Grado en Ingeniería en Informática

D. Jesús Enrique Sierra García, profesor del departamento de Ingeniería Civil, área de Lenguajes y Sistemas Informáticos

Expone:

Que el alumno D. Carmelo Basconcillos Reyero, con DNI 71290363T, ha realizado el Trabajo final del Grado en Ingeniería Informática titulado: Aplicabilidad de Gazebo en la simulación de vehículos autónomos.

y que dicho trabajo ha sido realizado por el alumno bajo la dirección del que suscribe, en virtud de lo cual, Se autoriza su presentación y defensa.

En Burgos a 3 de julio de 2017

D. Jesús Enrique Sierra García

Índice de contenido

Índice de ilustraciones

Índice de tablas

Resumen

La robótica ha causado un gran impacto en el mundo actual, tanto es así, que a ninguno nos extraña ver robots en nuestra vida cotidiana, ya sea un aspirador que nos limpia la casa y cuando queda poca batería vuelve a cargar solo, o bien un robot que nos ayude con las labores de cocina… De la misma manera que la robótica aparece en el sector servicios, los robots también están presentes en el sector de la industria, por lo que la empresa ASTI se dedica a crear elementos automatizados como vehículos de guiado automático facilitando así un transporte intralogistico flexible, eficiente y eficaz.

A estos vehículos de guiado automático se les denomina AGV (automatic guided vehicles). Su función es hacer más fácil el trabajo dentro de una empresa o almacén ya que se programan directamente y sus pruebas son realizadas sobre los mismos. Por contrapartida sus inconvenientes son:

- Falta de stock
- Daños en AGVs por una mala programación
- No disponibilidad de un entorno similar al deseado por el cliente

Es aquí donde nace nuestro proyecto, la solución a los problemas descritos serian solucionados con un testeo de estos robots en un simulador.

El objetivo de este proyecto es aprender a desarrollar un entorno que incluya la programación del robot, visualizando así los comportamientos de los AGVs en función a la planificación marcada. Todo esto con el fin de agilizar los plazos de aprendizaje por medio de un curso desarrollado por nosotros en *moodle.*

Para la comunicación con el robot, el sistema más extendido es ROS. En los últimos años este ha crecido llegando a incluir una gran comunidad de usuarios en todo el mundo. Históricamente la mayoría de los usuarios estaban en laboratorios de investigación, pero cada vez más vemos como su adopción en el sector comercial. Especialmente en robótica industrial y de servicios.

Palabras Clave

“Robótica, controlador PID, Gazebo, ROS, curso, SDF, modelado, ASTI”

Abstract

Robotics has caused a great impact in the world today, so much so, that we are not surprised to see robots in our daily life, whether a vacuum cleaner that cleans the house and when there is little battery recharges alone, or a Robot to help us with the kitchen work ... In the same way that robotics appears in the service sector, robots are also present in the industry sector, so the company ASTI is dedicated to creating automated elements as vehicles Automatic guidance facilitating flexible, efficient and effective intralogistic transport.

These automatic guiding vehicles are called AGV (automatic guided vehicles). Its function is to make work easier within a company or warehouse since they are programmed directly and their tests are carried out on them. On the other hand, its disadvantages are:

- *Lack of stock*
- *Damage to AGVs due to poor programming*
- *No availability of an environment similar to the one desired by the client*

It is here that our project is born, the solution to the problems described would be solved with a test of these robots in a simulator.

The objective of this project is to learn to develop an environment that includes the programming of the robot, thus visualizing the behaviors of the AGVs in function of the marked planning. All this in order to speed up the learning periods through a course developed by us in moodle.

For communication with the robot, the most widespread system is ROS. In recent years this has grown to include a large community of users around the world. Historically the majority of users were in research laboratories, but more and more we see as their adoption in the commercial sector. Especially in industrial robotics and services.

Key words

"Robotic, PID controller, Gazebo, ROS, course, SDF, modelling, ASTI"

I - INTRODUCCIÓN

Este proyecto es un trabajo que se realizará en colaboración con la empresa ASTI, concretamente con el tutor de mi proyecto Don Jesús Enrique Sierra Garcia, profesor de la Universidad de Burgos, y trabaja en el departamento de I+D de ASTI.

El problema planteado consiste en determinar si en dicha empresa sería viable realizar simulaciones en ordenadores sobre los robots que ellos mismos producen. Esto con el fin de determinar las ventajas o inconvenientes que puede producir su desarrollo.

En primer lugar deberiamos abordar el tema de la incrusión del robot dentro del propio simulador, para más tarde comunicarnos con él, y poder manejarlo de manera remota

Como solución se propone el simulador Gazebo. Es idóneo puesto que es gratuito, cuenta con un motor de renderizado avanzado, soporte para plugins, programación en la nube, un enorme repositorio con la mayoría de robots comerciales y una extensa gama de sensores y cámaras. Su principal ventaja es que se trata de un programa de uso gratuito y su comunidad es bastante grande, por lo que aparecen nuevos modelos de robots.

Realizaremos diferentes pruebas con varios robots para aprender el funcionamiento del programa, así como de la comunicación, para finalmente realizar un estudio sobre este sistema dentro de la empresa.

El objetivo es que de la misma manera que nos comunicamos con el robot que se encuentra dentro de nuestro simulador, en un futuro una vez comprobemos si es viable usar este simulador, podamos usarlo con uno real.

Para la comunicación con el robot usaremos un sistema operativo para robots denominado ROS. Es un sistema operativo de código abierto donde existe una comunicación (publicador/subscriptor) entre nodos, y de esta manera poder ejecutar ordenes.

Los nodos son programas ejecutables que realizan funciones concretas, por ejemplo acceso a un motor, mapeo del entorno, planificación de rutas, etc. Los nodos son compilados individualmente unos de otros, ejecutados y gestionados por un nodo principal (ROS Master).

Los nodos se comunican entre sí mediante el uso de mensajes, cada uno de estos está escrito usando una librería cliente de ROS la cual es una colección de código que facilita el trabajo a los programadores. Mediante el uso de esta librería podemos no solo crear los nodos sino también los publicadores y subscriptores de los mensajes, topics en los nodos, y sus parámetros.

Más tarde de manejar toda está tecnología deberemos ver la viabilidad que tiene implantar está tecnología en la empresa ASTI, y para concretar mejor los puntos importantes llevaremos a cabo un informe asesorando a dicha empresa sobre el uso de la simulación ROS-Gazebo en su empresa, así como de ser favorable un curso que recoja parte de la experiencia adquirida con este proyecto, y agilice a otras personas el aprendizaje de estas tecnologías.

II - OBJETIVOS DEL PROYECTO

En este apartado se explica de forma precisa y concisa cuales son los objetivos que se persiguen con la realización del proyecto. Se distingue entre los marcados por los requisitos del software a construir y los de carácter técnico que se plantean a la hora de llevar a la práctica el proyecto.

1. Objetivos

Se van a mostrar los puntos que se quieren tratar en este proyecto. Para ello el tutor del proyecto Don Jesús Enrique García de la empresa ASTI, nos plantea un problema a analizar sobre la aplicabilidad de gazebo-ROS en la compañía con el fin de evaluar su viabilidad. A grandes rasgos se podría resumir en dos objetivos principales:

- Realizar un informe final:

  En este informe deberemos valorar todo lo que hemos aprendido, evaluando sus posibles ventajas:

  - Analizar las características de la empresa y el impacto que podría tener la inclusión de este sistema en la misma
  - Hacer un balance económico
  - Aconsejar a la empresa sobre su implantación

- Realizar un curso que acorte el aprendizaje:

  El objetivo de este proyecto es que en el caso de que sea viable implantar esta tecnología, tras haber aprendido los primeros pasos en su utilización,
  seamos capaces de realizar un curso donde venga todo bien explicado y así acorte los plazos de aprendizaje para otras personas.

2. Resumen

En lineas generales las primeras semanas serán llevadas a cabo para documentarnos y aprender como funciona el simulador, realizaremos distintos tutoriales y pruebas, y decidiremos cuál será el entorno de trabajo con el que seguir. Una vez elegido ROS deberemos instalar todas sus librerías, aprender a manejar este complejo sistema operativo, lanzar un mundo con nuestro robot, que el robot esté en un nodo de ROS y lograr la comunicación con el robot de tal manera que interprete nuestras ordenes.

Finalmente es importante elaborar un informe en el cual saquemos una buena y estudiada recomendación para la empresa ASTI, con la que ellos puedan decidir si desean implantar dicha tecnología o no. Para facilitar el trabajo a otras personas que deseen aprender en este mundo ROS crearemos un curso para usuarios.

III - CONCEPTOS TEÓRICOS

Describiremos los conceptos básicos para poder entender nuestro proyecto.

1. Concepto básicos de Gazebo

Gazebo es el simulador que hemos elegido para desarrollar nuestro robot. Como muestra el siguiente gráfico, para diseñar en él debemos tener en cuenta las 3 dimensiones donde z, representa el alto, x representa el ancho e y representará la profundidad. En la siguiente ilustración se puede visualizar mejor. [1]

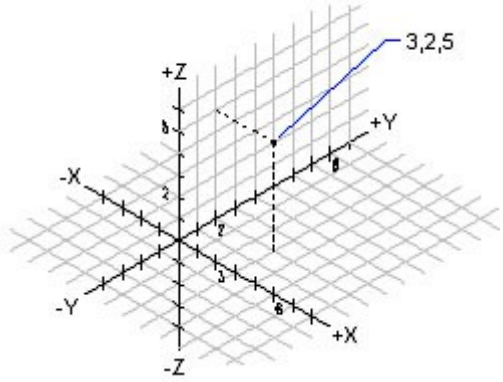

Ilustración 1: Eje coordenadas

Por otro lado a la hora de diseñar un robot en nuestro mundo, debemos tener en cuenta la gravedad ya que es indispensable crearla así como el aspecto luminoso de crear un sol en nuestro mundo que pueda dar sombras a los objetos. Estos objetos (ya sean robots, cuerpos inmóviles o cajas) tienen que tener unas respuestas físicas en nuestro mundo. Es necesario indicarlo como cuerpos rígidos para darle a entender que nuestros objetos no serán estáticos, si no que sufrirá modificaciones debido al impacto con otros y así también poder asignarle la masa, la gravedad, entre otros aspectos.

También es importante añadir una colisión a nuestro objeto para establecer cuales son los limites del objeto y a partir de que posición el objeto sufrirá un contacto. Por ejemplo si queremos crear una pared simplemente le ponemos una colisión y adaptamos los bordes al tamaño de la misma ya que esta no sufrirá cambio físico. En cambio si va a ser un objeto que pueda ser movido, habrá que marcarle una colisión y la masa del cuerpo.

Es importante optimizar a la hora de seleccionar las colisiones ya que para que estas funcionen, la computadora tiene que trabajar para hacer cálculos. Mientras mas objetos colisionen a la vez, mas trabajo tendrá por hacer, por lo que si se puede utilizar colisiones con menos partes del robot será muy eficaz para el rendimiento.

2. Concepto básicos de ROS

ROS se compone de un conjunto de librerías de programación, aplicaciones, drivers y herramientas de visualización, monitorización, simulación y análisis, todas reutilizables para el desarrollo de nuevas aplicaciones para robots tanto simulados como reales. Por otro lado, ROS nos proporciona también servicios estándares propios de un sistema operativo pero sin serlo, ya que se instala sobre otro. En general Linux y de manera recomendada Ubuntu, y por ello también recibe la denominación de Meta-Sistema Operativo. Posee su propio manejador de paquetes mediante comandos desde terminal para la gestión, compilación y ejecución de archivos, así como abstracción de hardware.

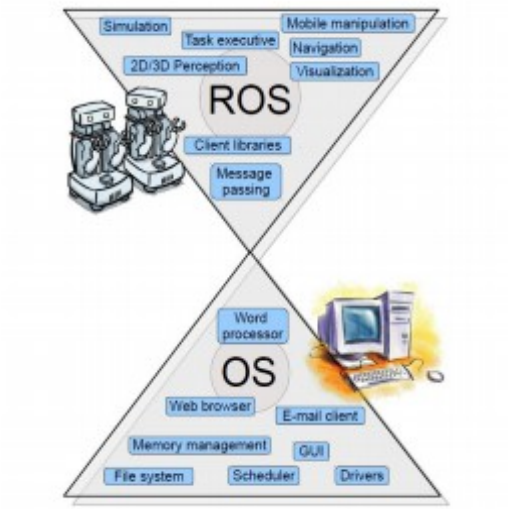

Ilustración 2: Relación ROS y Sistema operativo

Su estructura es modular, de forma que cada programa o aplicación (lo que en ROS se denomina como nodo) se ejecuta dentro de otro ejecutable (Máster) que funciona de núcleo del sistema y hace las veces de plataforma por la que se comunican los diferentes nodos mediante topics y/o servicios, de los que se hablará más adelante.

Máster: El Maestro proporciona registro de nombres y la búsqueda para el resto de los nodos. El maestro es clave para mandar mensajes entre nodos, intercambiar, o invocar los servicios, lo que hace que sea totalmente indispensable a la hora de ejecutar cualquier tipo de programa.

Nodos: Los nodos son procesos independientes que realizan la computación. Un sistema de control de robots comprende por lo general muchos nodos. Por ejemplo, un nodo controla un láser, un nodo controla los motores de las ruedas, un nodo lleva a cabo la localización, un nodo realiza planificación de trayectoria, un nodo proporciona una vista gráfica del sistema, y así sucesivamente. Un nodo de ROS se escribe con el uso de las librerías cliente de ROS, roscpp y rospy. La herramienta rosnode es una herramienta de línea de comando para mostrar información sobre los nodos:

rosnode info node: Muestra información sobre el nodo

rosnode kill node: Mata el nodo o envía una señal para matarlo

rosnode list: Muestra una lista con los nodos activos

rosnode machine hostname: Lista los nodos ejecutando en una máquina en concreto

rosnode ping node: Muestra la conectividad con el nodo

rosnode cleanup: Limpia la información de registro para nodos inalcanzables

Mensajes: Los nodos se comunican entre si por medio de mensajes. Un mensaje es una estructura de datos compuestos. El mensaje es combinación de tipos primitivos y mensajes. Los tipos primitivos de datos (integer, floating point, boolean, etc.) están soportados, como los arrays de tipo primitivo. Los mensajes pueden incluir estructuras y arrays (como las estructuras de C).

Tópicos: Los tópicos son canales de comunicación para transmitir datos. Los tópicos pueden ser transmitidos sin una comunicación directa entre nodos, significa que la producción y consumición de datos esta desacoplada. Un tópico puede tener varios subscriptores. Cada tópico esta fuertemente tipado por un tipo de mensaje de ROS que se publica, los nodos pueden recibir mensajes de un tipo determinado. Un nodo puede subscribirse a un tópico solo si tiene el mismo tipo de mensaje. El tópico en ROS puede transmitirse usando TCP/IP o UDP. El transporte basado en TCP/IP es conocido como TCPROS y utiliza TCP/IP para la conexión. Este es el tipo de transporte utilizado en ROS. ROS tiene una herramienta para trabajar con tópicos llamada rostopic. Es una herramienta de línea de comandos que proporciona información sobre el tópico o publica datos directamente sobre la red:

rostopic bw /topic: Muestra el ancho de banda utilizado por un tópico

rostopic echo /topic: Muestra el mensaje por la salida estándar.

rostopic find message_type: Busca tópicos que usen el tipo de mensaje especificado.

rostopic hz /topic: Publica la tasa de publicación del tópico.

rostopic info /topic: Muestra información sobre el tópico, el tópico publicado, los que están subscritos y los servicios.

rostopic list: Muestra información sobre los tópicos activos.

rostopic pub /topic type args: Publica datos al tópico. Permite crear y publicar datos en cualquier tópico directamente desde la línea de comandos.

rostopic type /topic: Muestra el tipo de tópico, es decir, el tipo de mensaje que publica.

Servicios: En muchos casos el transporte en un único sentido no es suficiente para las interacciones de petición y respuesta que a menudo se requieren en un sistema distribuido. La petición y respuesta se realiza a través de los servicios, que se definen a partir de una estructura de mensajes: una para la petición y otra para la respuesta. Un nodo proporciona un servicio con un nombre y un cliente utiliza dicho servicio mediante el envío del mensaje de petición y espera a la respuesta. La librería cliente de ROS generalmente presenta esta interacción como si fuera una llamada a procedimiento remoto. Los comandos soportados son los siguientes:

rosservice call /service args: Llama al servicio con los argumentos apropiados

rosservice find msg-type: Busca los servicios por el tipo de servicio

rosservice info /service: Muestra información sobre el servicio

rosservice list: Lista los servicios activos

rosservice type /service: Muestra el tipo de servicio

rosservice uri /service: Muestra el servicio ROSRPC URI

Para crear, modificar y trabajar con paquetes, ROS proporciona algunas herramientas para asistencia:

rospack: Este comando se usa para obtener información o buscar información sobre un paquete

roscreate-pkg: Crear un nuevo paquete utiliza este comando

rosmake: Compilar un paquete

rosdep: Instalar en el sistema las dependencias de un paquete

rxdeps: Ver las dependencias de un paquete como un grafo

Para moverte entre los paquetes y sus ficheros y archivos, ROS proporciona un paquete muy útil llamado rosbash, que proporciona algunos comandos muy similares a los de Linux.

roscd: Ayudar a cambiar de directorio

rosed: Editar un archivo

roscp: Copiar un fichero de algún paquete.

rosd: Listar los directorios de un paquete

rosls: Listar los ficheros de un paquete[2]

3. Controlador PID

Un controlador PID (proporcional-integral-derivativo) es un mecanismo de control por re-alimentación ampliamente usado en sistemas de control industrial. Este calcula la desviación o error entre un valor medido y un valor deseado. El controlador intenta minimizar el error a lo largo del tiempo.

En este proyecto se usa para hacer que el robot ajuste su movimiento lo más posible a una trayectoria prefijada, utilizando las variables PID para su regulación.

El algoritmo del control PID consiste de tres parámetros distintos: el proporcional, el integral, y el derivativo. El valor Proporcional depende del error actual. El Integral depende de los errores pasados y el Derivativo es una predicción de los errores futuros. La suma de estas tres acciones es usada para ajustar al proceso por medio de un elemento de control.

3.1. Proporcional

La parte proporcional consiste en el producto entre la señal de error y la constante proporcional para lograr que el error en estado estacionario se aproxime a cero, pero en la mayoría de los casos, estos valores solo serán óptimos en una determinada porción del rango total de control, siendo distintos los valores óptimos para cada porción del rango. En cambio, existe el fenómeno de la sobreoscilación que consiste en que el sistema alcanza valores superiores a los deseados. Hay una relación lineal continua entre el valor de la variable controlada y la posición del elemento final de control. La parte proporcional no considera el tiempo, por lo tanto, la mejor manera de solucionar el error permanente y hacer que el sistema contenga alguna componente que tenga en cuenta la variación respecto al tiempo, es incluyendo y configurando las acciones integral y derivativa.

La fórmula del proporcional está dada por:

$$P_{\text{sal}} = K_p\, e(t)$$

Ilustración 3: Fórmula del cálculo proporcional

El error, la banda proporcional y la posición inicial del elemento final de control se expresan en tanto por uno. Nos indicará la posición que pasará a ocupar el elemento final de control.

3.2. Integral

El modo de control Integral tiene como propósito disminuir y eliminar el error en estado estacionario, provocado por perturbaciones exteriores y los cuales no pueden ser corregidos por el control proporcional. El control integral solo actuará cuando hay una desviación entre la variable y el punto deseado, integrando esta desviación en el tiempo y sumándola a la acción proporcional. El error es integrado, lo cual tiene la función de promediarlo o sumarlo por un período determinado; luego es multiplicado por una constante Ki. Posteriormente, la respuesta integral es adicionada al modo Proporcional para formar el control P + I con el propósito de obtener una respuesta estable del sistema sin error estacionario.

El control integral se utiliza para obviar el inconveniente del offset (desviación permanente de la variable con respecto al punto de consigna) de la banda proporcional.

La fórmula de la integral está dada por:

$$I_{\text{sal}} = K_i \int_0^t e(\tau)\, d\tau$$

Ilustración 4: Fórmula del cálculo integral

3.3. Derivativo

La acción derivativa solo se produce cuando hay un cambio en el valor absoluto del error; en caso de que el error sea constante, solamente actúan los modos proporcional e integral.

El error es la desviación existente entre el punto de medida y el valor consigna, o "Set Point".

El objetivo de la acción derivativa es evitar que el error se incremente, para ello mantiene el error al mínimo corrigiéndolo proporcionalmente con la misma velocidad que se produce.

Se deriva con respecto al tiempo y se multiplica por una constante Kd y luego se suma a las señales anteriores (P+I). Es importante adaptar la respuesta de control a los cambios en el sistema ya que una mayor derivativa corresponde a un cambio más rápido y el controlador puede responder acordemente.

La fórmula del derivativo está dada por:

$$D_{\text{sal}} = K_d \frac{de}{dt}$$

Ilustración 5: Fórmula del cálculo derivativo

El control derivativo se caracteriza por el tiempo de acción derivada en minutos de anticipo.

El tiempo óptimo de acción derivativa es el que retorna la variable al punto de consigna con las mínimas oscilaciones.

- **Kp constante de proporcionalidad:** se puede ajustar como el valor de la ganancia del controlador o el porcentaje de banda proporcional.
- **Ki constante de integración:** indica la velocidad con la que se repite la acción proporcional.
- **Kd constante de derivación:** hace presente la respuesta de la acción proporcional duplicándola, sin esperar a que el error se duplique. El valor indicado por la constante de derivación es el lapso durante el cual se manifestará la acción proporcional correspondiente a 2 veces el error y después desaparecerá.

Tanto la acción Integral como la acción Derivativa, afectan a la ganancia dinámica del proceso. Mientras que la acción integral sirve para reducir el error estacionario, que existiría siempre si la constante Ki fuera nula.

La salida de estos tres términos, el proporcional, el integral, y el derivativo son sumados para calcular la salida del controlador PID[3]. Definiendo y (t) como la salida del controlador, la forma final del algoritmo del PID es:[4]

$$\mathrm{y(t)} = \mathrm{MV(t)} = K_p e(t) + K_i \int_0^t e(\tau)\,d\tau + K_d \frac{de}{dt}$$

Ilustración 6: Fórmula del cálculo del PID

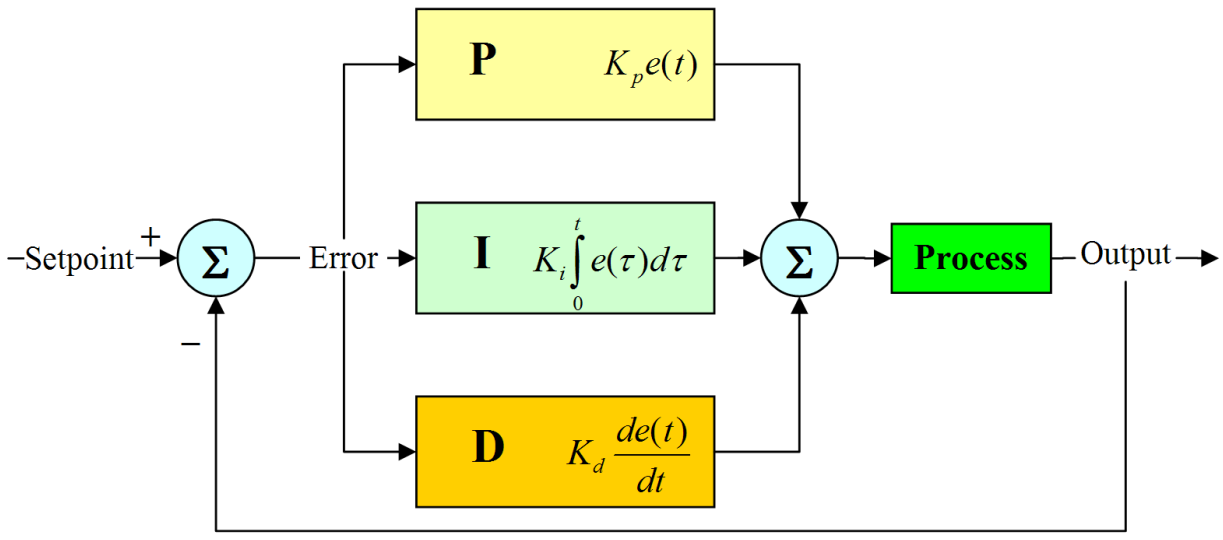

Ilustración 7: Esquema general de un controlador PID

IV - TÉCNICAS Y HERRAMIENTAS

En esta sección vamos a hablar de las distintas técnicas y herramientas que hemos usado y analizado, mostrando las razones que nos han influenciado para tomar la decisión de usarlo. Dada la gran cantidad de herramientas y técnicas para resolver los subproblemas de nuestro proyecto, hemos ido escogiendo varias de las herramientas disponibles y hemos elegido lo que mejor se ajustaba a nosotros. También trataremos los lenguajes usados.

Generalmente hemos optado por las opciones de programas vistos en la carrera, pero en algún caso la elección ha sido difícil, y hemos decidido por pequeños detalles.

1. SDF

Es un formato XML que describe objetos y entornos para simuladores de robots, visualización y control. Originalmente desarrollado como parte del simulador de robot Gazebo. A lo largo de los años, SDF se ha convertido en un formato estable, robusto y extensible capaz de describir todos los aspectos de robots, objetos estáticos y dinámicos, iluminación, terreno e incluso física.

Además de los atributos cinemáticos y dinámicos, se pueden definir sensores, propiedades superficiales, texturas, fricción de las articulaciones y muchas más propiedades para un robot. Estas características le permiten usar SDF para simulación, visualización, planificación de movimiento y control de robot.

Nosotros lo usaremos como lenguaje para desarrollar nuestros robots, o para modificar los creados ya en Gazebo. [5]

2. Gestor de tareas

2.1. Trello

Es un software de administración de proyectos con interfaz web en la cual podemos organizar nuestros proyectos de forma fácil e intuitiva desde cualquier equipo. En él que introduzcamos nuestra cuenta, nuestra información se guarda en la nube, por lo que podemos acceder desde cualquier equipo con Internet.

Ventajas:

- Como principal punto a favor diré que su curva de aprendizaje es corta, por lo que es fácil aprender a manejarse, además hemos tenido asignaturas en la carrera en las cuales hemos precisado de su uso

Desventajas:

- La principal desventaja por lo cual lo descartaremos es que no viene adjunto a nuestro repositorio, por lo que buscaremos otra solución que implemente esto.[6]

Podemos acceder a el a través del siguiente enlace: https://trello.com/

2.2. Wunderlist

Es una aplicación multiplataforma en el cual podemos gestionar todas las tareas de nuestros proyectos, cada tarea permite añadir un recordatorio, agregar subtareas, notas, o añadir comentarios para un trabajo corporativo.

Ventajas:

- La versión gratuita es muy completa y su manejo muy sencillo e intuitivo.
- Tiene muchas funciones pero a su vez cuentan con una buena usabilidad.

Desventajas:

- Su interfaz es un poco antigua, y necesitaría una actualización.[7]

Podemos acceder a el a través del siguiente enlace: https://www.wunderlist.com/es/

2.3. ZenHub

Es una herramienta que trabaja con tablones dinámicos, de la misma manera que lo hace trello (tablones Kanban), permite la subida de ficheros, y se puede incluir en nuestro repositorio Github.

Ventajas:

- Su principal ventaja es poder integrarlo en github, por lo que podremos trabajar en una sola aplicación.
- Esta ha sido la razón principal por la que he elegido este método.

Desventajas:

- No podremos añadir código pero tampoco es algo irremediable, ya que justo en el repositorio donde se integra la herramienta podemos visualizar dicho código.[8]

Podemos acceder a el a través del siguiente enlace: https://www.zenhub.com/

3. Gestores de versiones

3.1. GitHub

Es un repositorio de versiones donde vamos fijando nuevos issues(tareas a realizar) dentro de un milestone (tarea global a realizar) el código queda organizado en versiones, cada vez que hacemos un commit se actualizaran, y los elementos que modificados aparecen como tal.

Ventajas:

- Es de los repositorios mas usados y esta basado en Git.
- Muy útil para el seguimiento de proyectos, cualquiera que vea tu proyecto puede proponerte alternativas o nuevas soluciones.
- Hemos visto GitHub a lo largo de la carrera,por lo que facilita su manejo, es por eso que hemos elegido este.

Desventajas:

- Problemas cuando quieres actualizar más de 100 archivos desde la interfaz a veces da fallo.[9]

Podemos acceder a el a través del siguiente enlace: https://github.com/

3.2. GitKraken

Es un repositorio que funciona también con Git, su versión gratuita es bastante completa, tiene una interfaz muy vistosa y su manejo es rápido e intuitivo.

Ventajas:

- Tiene un práctico botón con para deshacer operaciones, se pueden revisar errores al momento, lo que hace más fluido el trabajo.
- Es un repositorio muy usado y que esta basado en Git que es casi la referencia en este tipo de proyectos.

Desventajas:

- No se puede incluir mas de veinte personas en los proyectos gratuitos.[10]

Podemos acceder a el a través del siguiente enlace: https://www.gitkraken.com/

4. Maquina virtual

4.1. VirtualBox

Es una herramienta de virtualización donde podemos crear maquinas virtuales donde instalamos el sistema operativo invitado dentro del sistema operativo de nuestro equipo.

Tiene un manejo fácil y nos permite modificar las características de nuestra maquina virtual después de ser creada.

Ventajas:

- Es propiedad de la empresa Oracle, pero de código abierto, por lo que es gratuita.
- La hemos usado durante la carrera en numerosas asignaturas.

Desventajas:

- Bastante lenta en ejecución.[11]

Podemos acceder a el a través del siguiente enlace: https://www.virtualbox.org/

4.2. VMWare

Es una herramienta de virtualización propiedad de Dell pero cuenta con una versión gratuita con una serie de limitaciones, su principal característica es que sus instrucciones se ejecutan sobre el hardware físico, mientras que otras lo hacen mediante la llamada al sistema operativo anfitrión.

Ventajas:

- Es más rápido que VirtualBox comparado en diferentes comparativas de Internet.
- Después de probar con ambos, sufría menos cuelgues al ejecutar programas de gran uso gráfico.
- Hemos elegido VMWare porque funciona mucho más rápido y sufre menos cuelgues, aunque inicialmente empezamos con VirtualBox por ser la más conocida por lo aprendido en clase.

Desventajas:

- La principal desventaja es que se trata de una versión gratuita, y cuenta con una serie de restricciones (como no poder cambiar el numero de procesadores a una maquina creada). [12]

Podemos acceder a el a través del siguiente enlace: http://www.vmware.com/

4.3. Simulador Gráfico

4.4. Gazebo

Es un simulador de múltiples robots para espacios abiertos, es capaz de simular conjuntos de robots, sensores y objetos, pero en un mundo tridimensional. Además reproduce de forma bastante real la reacción de sensores y la interacción entre dispositivos. Tienen modelos ya diseñados de robots y sensores con los que realizar las simulaciones.

Ventajas:

- Las conexiones entre los distintos dispositivos se realiza a través de direcciones
- IP asignándole un puerto a cada uno, de este modo se puede enviar señales a cada robot
- individualmente.
- Es un programa gratuito.
- Es el propuesto por el tutor, por eso lo cogemos.

Desventajas:

- No funciona correctamente en Windows.

Podemos acceder a el a través del siguiente enlace: http://gazebosim.org/

4.5. Roboworks

Es un simulador de la empresa Newtonium ,que se basa en la idea de la simulación tridimensional, y en proporcionar una interfaz3D a LabView, Matlab, y Visual Basic. Los modelos se generan gráficamente, y se controla su simulación por medio de archivos de datos, con el teclado, con la interfaz con LabView, Matlab o Visual Basic; o con Robotalk (software que nos permite, remotamente y desde cualquier terminal con TCP/IP, controlar la simulación.

Ventajas:

- Fácil utilización lo que constituye una de sus grandes bazas.
- Dispone de un completo tutorial en español.

Desventajas:

- Solo disponible para Windows 95/98/NT/2000.
- Su precio es de 540€.

Podemos acceder a el a través del siguiente enlace: http://www.newtonium.com/

4.6. Webots

Simulador de la empresa Cyberbotics, es permite la simulación de numerosos tipos de robots, incluidos robots definidos por el usuario. Incluye completas librerías con todo tipo de actuadores y sensores. Permite la programación del robot mediante C, C++ y Java.

Su precio varía según la licencia adquirida [3]: desde los 320 euros de la versión de educación, hasta los 3500 euros de la versión profesional. La diferencia entre ellos es que el primero carece de las siguientes características:

Ventajas

- Utiliza las librerías Open Dynamics Engine (ODE, que son de código abierto)
- para las simulaciones físicas, por lo que podremos modificarlas a nuestro gusto.
- Permite guardar las simulaciones en formato de vídeo AVI o MPEG.
- Incluye ejemplos
- Disponible para los sistemas operativos Linux, Windows, y Mac OS X.

Desventajas

- Su elevado precio desde los 320 euros de la versión de educación(versión reducida), hasta los 3450 euros de la versión profesional.[13]

Podemos acceder a el a través del siguiente enlace: https://www.cyberbotics.com/

Comparativa de los 3:

	Coste licencia	Robots integrados	Entorno editable	Diseñar propios robots	Lenguajes
Gazebo	Gratuito	Si	Si	Si	C, C++
Roboworks	540€	Si	No	Si	LavView, Matlab, VB
Webots	3500€(versión pro)	Si	Si	Si	C, C++ C#, VB#, J#

Tabla 1: Comparativa simuladores

5. IDE

Un IDE es una aplicación informática para poder desarrollar código y ejecutarlo. Facilita la navegabilidad por los paquetes y puedes hacer ejecuciones controladas de partes del código con puntos de rotura, donde mostrar los valores temporales de las variables(debugs). Estos son muy eficaces ,porque es mas sencillo corregir los fallos .

Se compone, normalmente, de un editor de código fuente y un depurador.

Una de las ventajas del IDE es que tiene también autocompletado, compilador, indentador de código e interprete.[14]

5.1. Eclipse

Es una aplicación software con muchos plugins y herramientas para el desarrollo de software. Está basado en Java para su ejecución e instalación y para ello es necesario tener Java instalado.

Fue desarrollado por IBM en un principio, pero ahora esta desarrollado por la Fundación Eclipse, una fundación independiente y sin animo de lucro.

Ventajas

- Identificar el código y editarlo con mayor facilidad
- Unificar todo el ciclo de desarrollo en una herramienta.
- Soporta múltiples lenguajes
- Lo hemos usado en múltiples asignaturas durante la carrera.
- Es gratuito por lo que podemos descargarlo sin problemas.

Desventajas

- En ocasiones da errores inexistentes y hay que reiniciar el equipo

Podemos acceder a la pagina a través del siguiente enlace: https://eclipse.org/home/index.php

6. NetBeans

Es un entorno de desarrollo integrado libre, consta con una gran comunidad en todo el mundo. Esta aplicación permite desarrollar aplicaciones a partir de componentes software llamados módulos.

Ventajas

- Es software libre, y gratuito sin restricciones de uso
- Funciona con menos fallos y más fluido que eclipse

Desventajas

- Esta hecho principalmente para java

Podemos encontrar la herramienta a través del siguiente enlace https://netbeans.org/

7. Modelado

El modelado consiste en la creación de diagramas que nos explica las relaciones que van a tener los objetos entre ellos dentro de nuestra aplicación.

La finalidad de los diagramas es presentar diversas perspectivas de un sistema, a las cuales se les conoce como modelo.

7.1. Astah

Es una herramienta de modelado UML para la creación de diagramas orientados a objetos. Podríamos hacerlos dibujando pero es mas preciso y profesional usar una herramienta que esta pensado para ello.

Ventajas

- Programa muy completo y sencillo de usar
- Fácil navegabilidad entre los diferentes diagramas
- Lo hemos usado en la carrera por lo que su uso es más sencillo
- Fáciles videotutoriales para aprender su manejo dentro de su pagina
- Hemos usado este programa por todas estas ventajas

Desventajas

- Su mayor desventaja es que su licencia cuesta 1190$ para un año, o 2490$ de manera continua, para hasta 10 trabajadores, aunque cuenta con la versión estudiante que usaremos.[15]

Podemos encontrar la herramienta a través del siguiente enlace http://astah.net/.

7.2. Dia

Es una aplicación informática para la creación de diagramas, desarrollada en el proyecto GNOME. El formato para leer y almacenar gráficos es XML. Puede producir salida en los formatos EPS, SVG y PNG.

Ventajas

- Programa gratuito
- Tiene un fácil aprendizaje, y sencilla interfaz
- Tiene cabida en Windows, Linux y OS X.

Desventajas

- Programa más simple, y menos completo que el anterior
- No podemos realizar todos los tipos de diagramas

Podemos acceder a la pagina a través del siguiente enlace http://dia-installer.de/index.html.es

8. Sistema Operativo para Robot

8.1. ROS

ROS (Robot Operating System) es una plataforma de desarrollo open source (bajo la licencia BSD) para sistemas robóticos. Proporciona toda una serie de servicios y librerías que simplifican considerablemente la creación de aplicaciones complejas para robots. ROS se compone de un numero de nodos independientes, cada uno se comunica con el resto de nodos utilizando el modelo publicador/subscriptor.

Ventajas

- Permite reutilizar código
- Sus amplias librerías agilizan el trabajo
- Paso de mensaje entre procesos
- Elegimos esta opción puesto que es la más extendida, la más conocida, y es de código abierto

Desventajas

- Curva de aprendizaje es muy lenta
- Solo disponible para Linux

Podemos acceder a la pagina a través del siguiente enlace: http://www.ros.org/

8.2. Microsoft Robotics Developer Studio 4

Es un entorno de programación basada en .NET libremente disponible para la creación de aplicaciones de robótica. Puede ser utilizado tanto por desarrolladores profesionales y no profesionales. Puede soportar una amplia gama de plataformas robóticas, ya sea corriendo directamente sobre la plataforma (si tiene un PC integrado con Windows) o controlar el robot desde un PC con Windows a través de un canal de comunicación, tales como Wi-Fi o Bluetooth.[16]

Ventajas

- Contiene una extensa documentación y una amplia serie de muestras y tutoriales
- Gran numero de robots compatibles

Desventajas

- Solo disponible para Windows
- Curva de aprendizaje es muy lenta

Para mas información podemos acceder a la pagina desde este enlace:
https://www.microsoft.com/en-us/download/details.aspx?id=29081

9. C++

C++ es un lenguaje de programación diseñado para extender al lenguaje de programación C mecanismos que permiten la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido. Usaremos este lenguaje porque es el que interpreta el sistema operativo ROS.[17]

10. Editor de texto

Un editor de texto que nos facilite el trabajo con el código.

10.1. Notepad

Esta herramienta simula un editor que nos colorea el código para su mejor interpretación. Tiene una instalación sencilla en ubuntu, y para ejecutarlo simplemente debemos escribir en una terminal, notebook, y nos abrirá el editor de texto.

Podemos encontrar la herramienta a través del siguiente enlace: https://notepad-plus-plus.org/

11. TortoiseSVN

TortoiseSVN es un software de control de versiones muy fácil de usar. Se basa en Apache Subversion (SVN). Proporciona una interfaz de usuario agradable y fácil para Subversión. Esta herramienta es libre y no tiene ningún tipo de restricción. Nos permite la sincronización de archivos y directorios y ha sido utilizada para mantener actualizado el repositorio donde se almacenaba el avance que se iba realizando en el proyecto.

12. Herramienta de escritura

12.1. OpenOffice Writer

Writer es el procesador de texto del paquete de herramientas de ofimática propiedad de Apache, denominado OpenOffice. Nos hemos decantado por esta por que la plantilla del proyecto venia en este formato, y sabemos usar el programa.

Ventajas

- Tiene la ventaja de que su descarga es gratuita.
- Existe una plantilla del proyecto adaptada en el formato propio de Writer (odt).
- Hemos aprendido a usar está herramienta en primero de carrera.

Desventajas

- No disponer de un plugin de Mendeley de forma fácil y accesible, para la bibliografía.

12.2. Microsoft Word

Se trata del procesador de textos del paquete de herramientas ofimáticas de Microsoft,

Microsoft Office. Es el más extendido en el mundo, y el a mi modo de ver más fácil de manejar.

Ventajas

- En este caso el software es de pago, pero se dispone de una licencia de Office 365 por ser alumno de la Universidad de Burgos.
- Sabemos manejarlo.

Desventajas

- No disponer de una plantilla del proyecto propia para su formato.

12.3. LaTeX

LaTeX es una alternativa de editor de texto bastante diferente a las opciones anteriores. Está basado en la inclusión de etiquetas en el texto que definen sus características. Este tipo de orientación ofrece una mayor libertad en el desarrollo de un documento si se aprende a usar de forma correcta.

Ventajas

- Dispone de una plantilla para el proyecto en formato propio.
- Ofrece una potencia mayor, y un mejor acabado de la memoria

Desventajas

- Deberíamos aprender a utilizarlo, por lo que usaremos Writter que ya sabemos trabajar con ello.

13. Herramienta para la gestión del curso

Describiremos las dos principales herramientas para crear un curso virtual.

13.1. Moddle

Moodle es la más utilizada en enseñanza en todo el mundo: en institutos, colegios, escuelas de negocios, universidades, etc. Lo bueno de Moodle es que la puedes instalar desde tu propio hosting con un solo clic.

Ventajas:

- Es gratuita
- Puedes elegir tu plantilla personalizada
- Hemos elegido esta por que es la más usada

Desventajas:

- La versión gratuita tiene mucha publicidad

13.2. Efront

Efront es mucho más nuevo y es una opción más avanzada orientada a usuarios un poco más exigentes que buscan un sistema diferente a Moodle. De él destacaría las estadísticas que tiene del progreso de los alumnos y que es un sistema donde el proceso de enseñanza está mucho más controlado.[18]

Ventajas:

- Funcionamiento es muy sencillo e intuitivo
- Optimizado solo con herramientas necesarias

Desventajas:

- Éste es más complejo de instalar y configurar que Moddle

14. Editor de presentaciones

14.1. PowerPoint

Es un programa de presentación desarrollado por la empresa Microsoft para sistemas operativos Windows y Mac OS. Es un programa diseñado para hacer presentaciones con texto esquematizado, así como presentaciones en diapositivas, animaciones de texto e imágenes prediseñadas o importadas desde imágenes de la computadora. Se le pueden aplicar distintos diseños de fuente, plantilla y animación. Este tipo de presentaciones suelen ser más prácticas que las de Microsoft Word.

Ventajas:

- Herramienta más usadas para presentaciones con diapositivas
- Funcionamiento sencillo e intuitivo
- Sabemos usar la herramienta

Desventajas:

- Es una herramienta de pago que viene dentro del paquete de Office

15. Grabador de vídeo

15.1. Presentation Tube

Programa que permite grabar la pantalla de nuestro ordenador a tiempo real mientras podemos interaccionar con el ratón diferentes opciones. Grabará también si lo deseamos el audio que introduzcamos por un micrófono.

Ventajas:

- Muy sencillo de usar
- Es gratuito

Desventajas:

- Solo exporta la versión gratuita en .asf y nos da error al reproducir

15.2. Moravi PTT converter

Software ideal para transformar presentaciones de PowerPoint en archivos de vídeo. Con este programa, podrá pasar fácilmente una presentación a MP4, AVI, WMV, MOV o cualquier otro formato de vídeo popular. También puede agregar voz o música a su presentación, optimizarla para un dispositivo móvil o prepararla para compartirla en línea.

Ventajas:

- Nos permite exportar a cualquier tipo de archivo
- Hemos seleccionado este porque el otro no funcionaba correctamente

Desventajas:

- Es un programa de pago (22,95€), tiene una versión gratuita por 30 días

V - ASPECTOS RELEVANTES DEL DESARROLLO DEL PROYECTO

En este capítulo vamos a destacar los puntos claves del proyecto y como se han llevado a cabo.

1. Lenguaje para el desarrollo del proyecto

Desde un primer momento se decidió el empleo de C++ como lenguaje central en la implementación del proyecto dado que permite trabajar con ROS. Aunque también hemos tenido que aprender a programar en SDF porque los modelados de robots se hacen con este lenguaje para Gazebo.

2. Herramienta para manejo de de robots empleado

Como ya hemos visto en el punto anterior, hay varias alternativas sobre qué herramienta usar para la comunicación con el robot. En este caso según lo propuesto por el Tutor, empezamos a aprender en el entorno ROS ya que tiene muchas ventajas como ser el más usado en la actualidad, estar en un crecimiento constante y ser completamente gratuito.

3. Manejo de SDF

Para poder programar bien los robots, conviene meter bastantes horas a probar los diferentes sensores (ya creados) que se pueden integrar a nuestro robot, probarles, y comprobar su funcionamiento. De estos es desde donde puedes sacar los datos para más tarde mediante ROS comunicarte con el, y poder ordenar movimientos en función de los parámetros recibidos de los sensores.

En este punto cabe destacar que manejar bien SDF, y insertar todos sus atributos aunque inicialmente no sean necesarios(ya sean masa, peso, su forma, inercia) es importante, porque más adelante vas a poder necesitar de estos para leer algún dato.

4. Pruebas experimentales ROS

Del proyecto realizado destacaría que para cada robot se vayan haciendo pruebas, de tal manera que en el publicador busques que el robot te diga sus partes o te escriba qué elementos tienen mas de una longitud, en definitiva hacer pruebas para ver como es cada robot.

De esta manera conseguimos que el robot *ver como* es cada parte, podríamos pedir que nos enseñe todas las partes de tipo joint(unión), y aplicarles fuerzas para ver sus movimientos, comprobar como se mueve es importante para comprender el manejo.

5. Controlador PID

Controlador que viene incluido por las librerías de gazebo, debemos usarlo para controlar la fuerza que ejercemos sobre el robot, el mismo hace mediciones y va retroalimentandose haciendo cálculos sobre su movimiento actual y el futuro para dar un movimiento correcto al robot.

6. Informe y Curso

Una vez finalizada la etapa de programación e investigación sobre estas tecnologías nos centraremos en realizar un informe detallado sobre la viabilidad de estas técnicas en una empresa de Robótica como puede ser ASTI, una vez realizado el informe se realizaría un curso con videotutoriales, presentaciones y material didáctico para ayudar a acortar los plazos de aprendizaje de esta tecnología.

VI - TRABAJOS RELACIONADOS

1. *Análisis de un brazo robótico con Gazebo y ROS para tareas de inspección remota en el CERN*

Este trabajo de Fin de Máster en Automática y Robótica de José Luis Samper Escudero fue hecho para la organización europea para la investigación Nuclear. Trata de mejorar las tareas de inspección y mantenimiento mediante la integración de un manipulador robótico de 6 grados de libertad que desarrollará en Gazebo_ROS, y realizará tareas de mantenimiento que comprenden desde simples inspecciones visuales a complejas labores como puede ser desatornillado o extracción de componentes dañados.

Este proyecto, pese a que la finalidad no es la misma que el mio, se desarrolla de una forma parecida, y los pasos a realizar son similares, usa una metodología similar a la nuestra.

Enlace al trabajo: http://oa.upm.es/39677/1/TFM_JOSE_LUIS_SAMPER_ESCUDERO.pdf

2. *Operación de remachado mediante robot manipulador industrial basado en ROS*

El trabajo de fin de grado en Ingeniería de las Tecnologías Industriales de Víctor Hugo Gómez Tejada de la Universidad de Sevilla se sitúa en un entorno de simulación donde debemos realizar una tarea de remachado a una pieza de trabajo mediante un robot manipulador industrial. Programa un robot mediante el framework robótico ROS para realizar la tarea de remachado, evaluando su rapidez y eficiencia.

Aunque desde el punto de vista de programación no está muy evolucionado, está muy bien definida toda la parte teórica de los ejes de coordenadas. Tiene un gran trabajo de documentación y explicaciones.

Enlace al trabajo: http://bibing.us.es/proyectos/abreproy/90360/fichero/TFG+V%C3%ADctor+Hugo+G%C3%B3mez+Tejada.pdf+

3. *Extendiendo las capacidades del robot RESCUER en ROS*

Este trabajo de Julio Jesús León Pérez del Máster universitario en Sistemas Inteligentes de la Universidad Jaume I trata de como un robot físico con el que cuentan en dicha universidad lo usan en distintas plataformas, para poder comprobar su comportamiento tanto real como virtual. En la parte virtual al probar con MoveIt lo hacen con éxito, puesto que la programación hecha da la respuesta esperada, pero en Gazebo les queda bastante trabajo por realizar puesto que el robot se mueve de manera errática.

Es un trabajo similar al nuestro en la parte de Gazebo, donde por medio de ROS implementan un robot creado y le dan movimiento.

Enlace al trabajo: http://repositori.uji.es/xmlui/bitstream/handle/10234/147985/TFM_Leo%CC%81n%20Pe%CC%81rez_Julio%20Jesu%CC%81s.pdf?sequence=1

VII - CONCLUSIONES Y LÍNEAS DE TRABAJO FUTURAS

En este apartado, vamos a mencionar algunas conclusiones a las que hemos llegado tras realizar el proyecto. Se verán las partes más importantes que se han tomado para que futuros desarrolladores puedan continuar el proyecto.

1. Conclusiones

Considero que el proyecto llevado a cabo es muy ambicioso ya que se han abordado múltiples herramientas las cuales desconocíamos y nos han servido para ampliar conocimientos.

En primer lugar nos pusimos a documentarnos sobre la herramienta Gazebo, comenzamos a realizar tutoriales y nos fuimos dando cuenta de que cuanto más aprendíamos se abrían más campos en los que abordar.

Llegados a este punto, vimos que la mejor opción para seguir desarrollando el proyecto era aprender ROS, un sistema operativo para robots que se perfila como una de las mejores opciones en cuanto a la programación robótica.

Se abría de nuevo un nuevo mundo del que no sabíamos nada, empezábamos nuevamente a realizar tutoriales, y aprender de manuales, pero con la gran diferencia, que en el mundo ROS estaba mucho peor documentado.

Para plasmar el proyecto decidimos que la mejor forma era hacer un informe para una empresa de logística interna como ASTI y realizar un curso para acelerar el aprendizaje de terceros.

En general, podemos decir que implantar Gazebo-ROS en las empresas de logística es claramente beneficioso puesto que podemos simular un mundo similar a la realidad y evitar cometer fallos que en la realidad podrían ser fatales, de esta forma que una buena programación de un robot simulado puede servirnos de la misma manera para uno real, por lo que todo serían ventajas.

El sistema ROS se encuentra en constante desarrollo y evolución, por lo que el adecuado estudio del mismo no acaba con este proyecto, sino que representa una tarea permanente.

Los conocimientos adquiridos durante la realización de la carrera universitaria, me han ayudado de gran manera a llevar a cabo el proyecto. Las principales asignaturas sobre las que me he apoyado son:

- Sistemas Inteligentes: Sin duda la asignatura con la que guarda más relación dado que esta asignatura, nos introdujo en los entornos inteligentes, nos enseño a resolver problemas de IA con robots.
- Sistemas Operativos: La navegación por Ubuntu mediante líneas de comandos, así como el manejo de diferentes terminales son conocimientos vistos en la asignatura.
- Gestión de proyectos: Sin duda otra de las asignaturas fundamentales en la planificación y gestión del proyecto, las metodologías ágiles han sido necesarias.

2. Lineas de trabajo Futuras

Como futuros proyectos propuestos se podría dividir en dos vertientes, una mas teórica, en la cual se pueden desarrollar robots en los entornos virtuales descritos en este documento, e incluso diseñar un robot similar a los generados en una empresa de logística interna, para comprobar que todo el programa funcionan de forma correcta,y una vertiente real, donde se vaya utilizando ROS para generar el código necesario para mover un robot real, al que se le pueden ir añadiendo mas funciones con el paso del tiempo. Por supuesto estas dos vertientes pueden ser complementaria, ya que lo ideal sería generar un robot en el sistema virtual, probar que todo el código es completamente funcional, y luego cargarlo en nuestro robot real.

2.1. Robot en entorno virtual

Diseñar el propio robot de la empresa, con sus mismas características físicas, y visuales, de tal manera que su comportamiento en el simulador debería de ser similar al de un robot real, deberíamos incorporarle los mismos sensores que tiene el robot real, y sus trayectorias ante las mismas fuerzas debería de ser igual.

2.2. Robot en un entorno real

Utilizar ROS en un robot ya creado de la galería que dispone Gazebo, y generar un código válido para poder mover un robot real, más tarde se podría entrar a manejar sus sensores también desde ROS, y así hasta tener el robot real completamente manejado por el sistema para robots ROS.

VIII - REFERENCIAS

Bibliografía

[1] Alexander Subirós Martínez , Aprende autocad 3D, 2006, http://www.mailxmail.com/curso-aprende-autocad-3d/introducir-coordenadas-z
2: Ricardo Ragel de la Torre, Entorno de simulaciónROS/Gazebo, 5 de febrero del 2017, 11:53 UTC
[3] colaboradores de la wikipedia, Controlador PID, 10 de junio del 2017, 09:47 UTC
[4] Iñigo Gútiez, Ejemplo de control de temperatura, 1 de julio del 2017, 10:04 UTC, https://programacionsiemens.com/pid-en-step7/
[5] , API SDF, 14 de marzo del 2017, 10:03 UTC
[6] Fernando Siles, ZenHub.io, vitaminas para tu GitHub, 30 de mayo del 2017, 10:53 UTC, https://www.genbetadev.com/herramientas/zunhub-io-vitaminas-para-tu-github
[7] Miriam Schuager, Wunderlist estrena diseño y nuevas características en su versión web, , https://wwwhatsnew.com/2015/08/14/wunderlist-estrena-nuevo-diseno-en-su-version-web/
[8] Juan Diego Polo, zenhub, plataforma de gestión de proyectos integrada en Github, ahora gratis para estudiantes, , https://wwwhatsnew.com/2015/09/24/zenhub-plataforma-de-gestion-de-proyectos-integrada-en-github-ahora-gratis-para-estudiantes/
[9] colaboradores de Wikipedia, GitHub, 30 de junio del 2017, 11:44 UTC, https://es.wikipedia.org/w/index.php?title=GitHub&oldid=98817555
[10] Enric Florit, GitKraken, un nuevo cliente de git multiplataforma y gratuito, 30 de abril del 2017, 11:48 UTC, http://www.enricflorit.com/gitkraken-un-nuevo-cliente-de-git-multiplataforma-y-gratuito/
[11] colaboradores de Wikipedia, VirtualBox, 3 de febrero del 2017, 12:06 UTC
[12] colaboradores de Wikipedia, VMWare, 4 de Febrero del 2017, 12:08 UTC
[13] Elena Rodrigo López Javier Puertas Torres Diego López García Ramón García Olivares-Francisco Javier Fernández Oscar Yago Corral Guillermo Asín PrietoEsther Samper Domeque, Simuladores en robótica, 3 de mayo del 2017, 16:08 UTC
[14] Alida Vergara, Netbeans vs Eclipse ¿Cuál elegir?, 13 de marzo del 2017, 12:17 UTC, https://www.facilcloud.com/noticias/netbeans-o-eclipse-cual-elegir/
[15] colaboradores de Wikipedia, Astah, 30 de junio del 2017, 16:17 UTC
[16] Microsoft, Microsoft Robotics Developer Studio, 17 de marzo del 2017, 12:17 UTC, https://www.microsoft.com/en-us/download/details.aspx?id=29081
[17] colaboradores de Wikipedia, C++, 22 de abril del 2017, 12:23 UTC
[18] Javier Prieto Pariente, Estudio comparativo de herramientas de aprendizaje, 20 June 2017 12:27UTC, https://es.scribd.com/doc/34158569/Comparacion-entre-distintas-plataformas-e-learning-Moodle-Dokeos-Caroline-ATutor-y-EFront

Impreso en Burgos el lunes, 3 de julio de 2017